Львівський національний університет імені Івана Франка

На правах рукопису

Волощук Орест Романович

УДК 004.853+004.855.5

Методи кластеризації на великих масивах даних

01.05.03 — математичне та програмне забезпечення обчислювальних машин і систем

Магістерська робота

Науковий керівник

Годич Олесь Васильович,

кандидат фізико-математичних наук, доцент

Львів — 2011

ЗМІСТ

ВСТУП

Актуальність теми. Сьогодні все частіше виникають задачі, так чи інакше пов'язані із розбиттям масиву об'єктів на групи за певними критеріями. Із розвитком обчислювальної техніки збільшуються також об'єми баз даних, що можуть піддаватись такому аналізу. Виникає необхідність створення оптимальних алгоритмів обробки великих масивів даних.

Мета і завдання дослідження. Метою дослідження є розвиток методики кластеризації великих масивів даних. Для досягнення цієї мети були сформульовані та вирішені такі основні завдання:

— провести детальний аналіз та дослідити ефективність різноманітних алгоритмів кластеризації;

— здійснити контроль якості результатів роботи алгоритмів;

— розробити методи та виявити можливі оптимізації, що дозволять прискорити виконання задачі кластеризації

Об'єкт дослідження. Об'єктом дослідження є методи ієрархічні та плоскі алгоритми кластеризації даних.

Предмет дослідження. Предметом дослідження є придатність алгоритмів кластеризації до використання на великих об'ємах даних.

РОЗДІЛ 1
ОГЛЯД СТАНУ ПРОБЛЕМИ ТА ОСНОВНІ ПОНЯТТЯ

1.1. Історія кластеризації

Термін „кластерний аналіз", вперше використаний Тріоном у [15], означає набір підходів та алгоритмів, призначених для об'єднання схожих об'єктів у групи. Ця технологія знайшла своє застосування в цілій низці галузей наук та є необхідною частиною більшості сучасних засобів аналізу даних.

В 1959 радянський вчений Терентьєв розробив так званий „метод кореляційних плеяд" [17], покликаний здійснювати групування на базі корелюючих ознак об'єктів. Займаючись вивченням кореляцій між різними ознаками озерної жаби, він об'єднав їх в групи за абсолютною величиною коефіцієнту кореляції. Таким чином він отримав дві групи ознак – ознаки із великим та з малим значенням кореляції. Терентьєв назвав ці групи „кореляційними плеядами" та опублікував декілька методів їх аналізу. Це посприяло розвитку методів кластеризації за допомогою графів.

На початку 50х років також вийшли публікації Р. Льюїса, Е. Фікса та Дж. Ходжеса, присвячені ієрархічним алгоритмам кластеризації. Відчутний поштовх технології кластерного аналізу дали роботи Розенблатта про розпізнаючий пристій „перцептрон", котрі поклали початок теорії „розпізнавання без вчителя". Поштовхом до розробки методів кластеризації стала публікація [11] в 1963 році. В своїй роботі автори виходили з того, що для створення ефективних біологічних класифікацій процедура кластеризації повинна використовувати всеможливі показники, що характеризують досліджувані організми, проводити оцінку ступеня схожості між цими ор-

ганізмами, та забезпечувати розташування схожих організмів в одну групу. При цьому сформовані групи повинні бути досить локальними, тобто схожість організмів всередині групи повинна бути більшою, ніж між групами. Подальший аналіз таких груп допоможе вияснити, чи відповідають вони реальним біологічним класифікаціям. Сокел та Сніт, автори цеї роботи, вважали, що виявлення структури розподілу об'єктів у групи допоможе встановити процес утворення цих груп.

В середині сімдесятих з'явилась також низка робіт, присвячених методам аналізу якості здійсненої кластеризації. Індекси Данна [5], Девіса-Боулдіна [3] певною мірою дозволяють оцінити придатність результатів таксономії до реального використання.

в цей період розвитку технології ще не приділялось великої уваги необхідності передбачення можливості масштабування алгоритмів. Вибірки, що піддавались кластеризації, не були настільки великими, щоб масштабування мало зміст.

1.2. Типові задачі

Із кластерів, що повертаються в результаті роботи алгоритмів таксономії, можна виділити типових представників , і надалі працювати не з кожним об'єктом великого масиву, а лише з цими представниками. Це дозволяє суттєво спростити аналіз даних. Кластеризація часто стає складовою якоїсь більшої задачі, інструментом підготовки даних для її розв'язання. Такі задачі завжди пов'язані із пошуком та виділенням змістовних структур із великих масивів даних.

До них можна віднести задачі сегментування та розпізнавання зображень, мовлення, пошуку прихованих закономірностей в даних. На практиці такі задачі виникають при розв'язуванні проблем, що виникають в медицині, соціології, економіці та низці інших сфер діяльності людини. меди-

цині, наприклад, техніка сегментування зображення дозволяє виділяти на томограмах окремі області і на підставі їх форми та забарвлення приймати ставити діагноз. В біології використовуються техніки кластеризації для виявлення взаємопов'язаних груп генів та їх впливу на живі організми. Кластеризація успішно застосовується у маркетингових дослідженнях для виявлення зв'язків між різними групами споживачів та потенційних покупців, цільових аудиторій, та оптимального позиціонування нової продукції. В соціологічних дослідженнях використовується кластеризація даних, отриманих з різних джерел, для спрощення їх подальшого аналізу.

У своїй праці [16] Загоруйко описує одну із таких задач. Новосибірські вчені вивчали причини переселення людей з сіл в міста. Були вислані експедиції в навколишні села, жителям яких задавали приблизно сто анкетних питань, що стосувались віку, сімейного становища, освіти та ін. Після завершення опитування дослідники постали перед необхідністю аналізувати більш ніж сім тисяч анкет, що містили понад сто питань кожна. Ці дані було введено в програму таксономії, котра повернула сім великих таксонів, середні характеристики яких дозволили дати зібраним даним змістовну інтерпретацію. Наприклад, виділився кластер, що містив переважно жінок середнього віку, котрі мали дорослих дітей в місті. Очевидно, представниці цього таксону, названого дослідниками „бабусі", їхали в місто доглядати за своїми внуками. Решту таксонів опрацьовано аналогічно.

Перед процедурою кластеризації часто дані буває необхідно підготувати. В практиці нормою є випадки, коли для деяких об'єктів бракує частини атрибутів – в такому разі перед кластеризацією необхідно здійснити передбачення цих атрибутів, користуючись наявною інформацією. Також значення атрибутів об'єктів необхідно нормувати.

1.3. Основні поняття та означення

Дамо математичне означення кластеризації. Нехай D – множина об'єктів.

Означення 1.1. *Кластеризацією* $C = \{C \mid C \subseteq D\}$ називається таке розбиття D на множини, для якого виконується $\cup_{C_i \in C} = D$ і $\forall C_i, C_j \in C$: $C_i \cap C_{j \neq i} = \varnothing$. Множини C_i називаються кластерами.

Як вже згадувалось раніше, задача кластеризації полягає в знаходженні такого розбиття C, щоб схожі між собою об'єкти належали до одного кластера, а не схожі - до різних. Необхідно визначити спосіб обчислення схожості об'єктів. Для цього на просторі об'єктів вводиться певна метрика, геометричний зміст якої – відстань між об'єктами. Вона використовується як величина, обернена до міри схожості між об'єктами. На даний момент розроблено і широко застосовується наступний набір метрик:

— евклідова відстань

$$\rho(x, x') = \sqrt{\sum_{i=1}^{n} (x_i - x'_i)^2}$$

— квадрат евклідової відстані

$$\rho(x, x') = \sum_{i=1}^{n} (x_i - x'_i)$$

(використовується для надання більшої ваги об'єктам, розташованим далеко один від одного)

— манхеттенська відстань

$$\rho(x, x') = \sum_{i=1}^{n} \mid x_i - x'_i \mid$$

— відстань Чебишева

$$\rho(x, x') = max(\mid x_i - x'_i \mid)$$

ця метрика дозволяє розрізнити об'єкти, якщо вони відрізняються лише одною координатою

1.4. Алгоритми кластеризації

На даний момент значного розвитку набула ціла низка різноманітних алгоритмів кластеризації. Їх можна розділити на групи за різними ознаками:

— спосіб групування
  - плоскі
  - ієрархічні

— визначеність
  - визначені
  - невизначені

— чутливість до форми кластерів
  - кластери мають строго сферичну форму
  - кластери можуть мати довільну форму

Плоскі алгоритми будують одне розбиття вибірки на кластери. На відміну від них, агломеративні будують цілу систему взаємовкладених кластерів. На виході алгоритму отримується дерево кластерів, коренем якого служить кластер, що містить усі об'єкти вибірки, а листками є найменші кластери з одним об'єктом кожен. В такому разі дослідник має можливість обрати такий переріз дерева, що найкраще відповідатиме його потребам.

Визначені алгоритми однозначно ставлять у відповідність кожному об'єкту один кластер. Невизначені не дають такої чіткої інформації. Замість

ідентифікатора кластера вони повертають імовірність, із якою об'єкт належить до кожного із кластерів.

На сьогоднішній день широко використовуються наступні алгоритми кластеризації:

— k-means ([14], [8])
— UPGMA ([12])
— DBScan
— FOREL ([16])
— c-means
— карти Кохонена ([10])

РОЗДІЛ 2
АНАЛІЗ АЛГОРИТМІВ

В даній роботі розглядається ефективність чотирьох алгоритмів кластеризації. Два із них плоскі: k-means та DBSCAN. Інші два ієрархічні: UPGMA та споріднений із ним алгоритм пошуку найближчого сусіда.

2.1. Опис алгоритмів

K-Means. K-means — це метод кластеризації, що повертає таке розбиття, в якому кожен об'єкт належить кластеру із найближчим центром. Задача побудови такого розбиття належить до класу NP [9], тому на практиці застосовуються евристичні реалізації. Результатом роботи k-means є плоске розбиття. Для роботи алгоритму потрібно заздалегідь володіти інформацією про кількість кластерів, на які розбивається вибірка.

Оскільки така реалізація є евристичною, результатом її роботи може стати не глобальне, а локальне розбиття. Зокрема, вони залежать від вибору первинних центроїд кластерів на етапі ініціалізації. До цього моменту є декілька відомих підходів, описаних зокрема в [7]. Можливим варіантом є вибір у якості центроїд випадкових об'єктів із вибірки. Інший варіант — призначення кожного об'єкта у випадковий кластер та вибір центрів цих кластерів як початкових центроїд.

Стандартною практикою при застосуванні k-means є кількаразовий запуск алгоритму із різними вхідними даними та вибір того результату, який повторюється найчастіше.

Для деяких наборів даних k-means збігається за експоненціальний час [2], але такі набори на практиці не зустрічаються. Згладжена складність

Алгоритм 1 Алгоритм k-means

Ініціалізація. Виберемо у довільний спосіб k точок $c_1^{(1)}, c_2^{(1)}, ..., c_k^{(1)}$, що стануть центроїдами кластерів. Номер ітерації $t := 1$.

1. Заповнюємо кожен із k кластерів тими об'єктами, для яких його центроїда є найближчою з-поміж центроїд усіх кластерів:

$$S_i^{(t)} = \{x_j : ||x_j - c_i^{(t)}|| \leqslant ||x_j - c_{c^*}^{(t)}||, i^* = 1, 2, ..., k\}$$

2. Порівняємо розподіл об'єктів по кластерах із отриманим на попередньому кроці. Якщо вони збігаються, алгоритм завершено. В іншому разі, переходимо до кроку 3.
3. Обчислимо нові центроїди кластерів $c_1^{(t+1)}, c_2^{(t+1)}, ..., c_k^{(t+1)}$:

$$c_i^{(t+1)} = \frac{1}{S_i^{(t)}} \sum_{x_j \in S_i^{(t)}} x_j$$

Збільшуємо номер ітерації: $t := t + 1$. Переходимо до кроку 1.

виконання такого алгоритму — поліноміальна [1].

Цей метод має суттєві недоліки. Один із найважливіших — необхідність задавати кількість кластерів перед запуском. Неправильно обрана кількість кластерів може призвести до некоректного розбиття. Тому після кожного виконання алгоритму потрібно проводити діагностичні перевірки. Крім того, алгоритм розрахований лише на кластери сферичної форми та приблизно однакового розміру. При аналізі вибірки, що містить одне велике скупчення та декілька маленьких, k-means схильний розколювати велике скупчення.

UPGMA. Unweighted Pair Group Method with Arithmetic Mean — алгоритм ієрархічної кластеризації, призначений для побудови філогенетичних дерев. Такі дерева відображають еволюційні взаємозв'язки між різними видами або іншими сутностями, що мають спільного предка.

на початку роботи алгоритму створюється набір кластерів, кожен із яких містить по одному об'єкту вибірки. На кожному кроці здійснюється

об'єднання двох найближчих кластерів у один. За відстань між кластерами береться середня відстань між усіма парами їх об'єктів, тобто:

$$d(A,B) = \frac{1}{\mid A \mid\mid B \mid} \sum_{x \in A} \sum_{y \in b} d(x,y)$$

Після завершення роботи алгоритму отримується один кластер, який містить усі об'єкти вибірки. Процес об'єднання кластерів під час роботи алгоритму можна зобразити у вигляді дерева. Воно ілюструє зв'язки між кластерами.

UPGMA належить до групи алгоритмів GCP (Global Closest Pair). На кожній ітерації необхідно знайти пару кластерів, найближчих одне до одного. Для цього необхідно мати доступ до матриці відстаней між усіма об'єктами. Очевидно, для досить великих масивів даних обчислення такої матриці займає багато часу, а кешування її потребує великих затрат пам'яті. Використання спеціальної структури даних для зберігання відстаней між об'єктами дозволило реалізувати алгоритм зі складністю $O(n^2)$ [6] відносно часу, що є очевидною нижньою межею для цього класу алгоритмів.

DBSCAN. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) — алгоритм плоскої кластеризації, розроблений у 1996 [4]. Це один із найчастіше згадуваних та використовуваних алгоритмів. DBSCAN здійснює розбиття, грунтуючись на густині розташування об'єктів вибірки, а не на відстані до центру. Це дозволяє виділяти кластери довільної форми. Алгоритм розрахований на роботу із даними, що містять певну кількість шуму. Деякі об'єкти вибірки можуть залишились без кластера. Алгоритм не потребує інформації про кількість кластерів.

DBSCAN приймає на вхід два параметри — ϵ (окіл, в якому слід шукати сусідів точки) та n (мінімальна кількість сусідів, необхідна для того, щоб точку не визнали шумом). DBSCAN аналізує усі точки вибірки. Ті з них, в

ϵ-околі яких є більш ніж n точок, додаються у кластери. Решта визнаються шумом.

Алгоритм 2 Алгоритм DBSCAN

Ініціалізація Оголосимо лічильник кластерів i рівним нулю.

1. Виберемо довільну точку p серед досі не відвіданих. Позначимо p як відвідану.
2. Знайдемо всіх її сусідів, що лежать в межах ϵ-околу p:
$$N = \{x : d(p, x) < \epsilon\}$$
3. Якщо $\mid N \mid > n$, створюємо новий кластер, додаємо до нього p. Інакше переходимо до кроку 1.
4. Вибираємо довільну точку p' серед невідвіданих точок із N. Позначаємо її як відвідану.
5. Шукаємо всіх її сусідів:
$$N' = \{x : d(p', x) < \epsilon\}$$
6. Якщо $\mid N' \mid > n$, додаємо p' до кластера та доповнюємо N:
$$N = N \cup N'$$
7. Якщо у N залишились невідвідані точки, переходимо до кроку 4, інакше закриваємо кластер та переходимо до кроку 1.

DBSCAN на кожній ітерації шукає найближчих сусідів одної з точок вибірки. При прямій реалізації, що передбачає обчислення усього рядка матриці відстаней на кожній ітерації, загальна складність алгоритму становить $O(n^2)$. При використанні індексованої структури даних для збереження матриці відстаней (наприклад, R-дерева, яке дозволяє виконати пошук сусідів в межах деякого околу за $(O(logn)))$, загальна складність алгоритму становитиме $O(nlogn)$. При використанні неоптимізованої структури даних складність становитиме $O(n^2)$. Щоправда, потреби у пам'яті також виростуть до $O(n^2)$.

Neighbor Joining. Це алгоритм ієрархічної кластеризації. Він споріднений із UPGMA, але між ними є суттєві відмінності. Якщо у випадку

UPGMA на кожній ітерації ми шукаємо глобально найближчу пару кластерів, то при neighbor joining здіснюється пошук та об'єднання кластерів, які є локально найближчими сусідами одне одного. Для цього достатньо зберігати повний ланцюжок найближчих сусідів. Такий підхід дає можливість вагомо зменшити кількість обчислень. Складність алгоритму визначатиметься кількістю пошуків найближчих сусідів.

Означення 2.1. Ланцюгом найближчих сусідів називається такий кортеж $P = (C_1, C_2, ..., C_N)$, в якому C_{i+1} є найближчим сусідом C_i.

Означення 2.2. Ланцюг найближчих сусідів називається повним, якщо він містить хоча б два кластери, і два останні кластери є взаємно найближчими сусідами.

На початку роботи алгоритму вибірка розбивається на кластери за правилом "один об'єкт — один кластер". Після цього будується повний ланцюг найближчих сусідів. Після цього останні два кластери об'єднуються в один і вилучаються із ланцюга найближчих сусідів, і побудова повного ланцюга найближчих сусідів продовжується. Якщо ланцюг спорожнів, починаємо будувати його знову із довільним чином вибраного кластера.

Додавання одного кластера до ланцюга і перевірка повноти ланцюга має часову складність $O(n)$. Ця операція здійснюється не більш ніж $2(n-1)$ разів. Таким чином, загальна часова складність алгоритму дорівнює $O(n^2)$.

2.2. Тестові дані

Ефективність алгоритмів перевірялась на даних Data Mining Cup 2008 року. Це інформація про один із турів німецької державної лотереї. Кожен об'єкт вибірки відображає учасника лотереї. Був проведений збір інформації про учасників лотереї — зокрема, сімейне та фінансове становище, освіта, рід діяльності та ін. Кількість об'єктів у вибірці — 113477. Кожен об'єкт вибірки має унікальний числовий ідентифікатор.

Для деяких об'єктів вибірки бракує даних. Для заповнення порожніх клітинок бази було застосовано модифікацію алгоритму ZET, описаного в [16]. В оригіналі цей алгоритм здійснює заповнення порожніх клітинок, користуючись інформацією про об'єкти, схожі на той, що містить порожню клітинку. Спочатку вибираються такі рядки, тоді серед цих рядків вибираються стовпці із найвищим коефіцієнтом лінійної регресії відносно атрибута, який потрібно передбачити. В кінці ітерації розв'язується рівняння лінійної регресії, коренем якого є шукане значення атрибута.

Я використав спрощену версію алгоритму, що вибирає значення атрибута як середнє значення у стовпці серед об'єктів, близьких до даного.

Для перевірки алгоритмів також використовувались синтетичні набори даних. Я генерував їх за допомогою утиліти, написаної спеціально із цею методю. Утиліта зчитує із вказаного файла послідовно інформацію про точку, яка буде центром групи об'єктів, та дозволені відхилення від центру по кожній координаті, і генерує вказану кількість об'єктів, розташованих випадковим чином у вказаних межах навколо центра. Задавши достатню кількість центральних точок та підібравши вдалі відхилення, можна добитись набору даних довільної форми. Зокрема, задавши центр угруповання приблизно посередині між усіма іншими точками, і досить великі відхилення по усіх координатах, можна досягнути ефекту шуму в даних. Як і основна програма, утиліта написана на C++.

2.3. Реалізація

Для перевірки ефективності вищеописаних алгоритмів була створена їх програмна реалізація. У виборі мови програмування я керувався міркуваннями швидкодії та можливості контролю ресурсів комп'ютера. Вибір зупинився на C++. Реалізація являє собою консольну програму без графічного інтерфейсу, що зчитує дані з файла та зберігає результати кластеризації

також у файлі.

Створено об'єктно-орієнтовану модель для зберігання об'єктів вибірки. Для зберігання власне даних я використовував стандартні колекції STL. Де-юре STL не належить до стандарту мови C++, але де-факто усі сучасні поширені компілятори мають підтримку цеї бібліотеки. Перша версія програми включала об'єкт „DataContainer", який містив асоціативний контейнер std::map, що дозволяв звертатись до обєктів вибірки по ідентифікатору. Кожен елемент вибірки був об'єктом типу Object, що містив std::vector вказівників на об'єкти класу Attribute. Кожен Attribute містив числове значення відповідного атрибута та інформацію при присутність даного атрибута у відповідного об'єкта вибірки.

Реалізовано також абстрактний клас AbstractMetric, який відповідав метриці простору. За задумом можна було реалізувати довільну метрику простору, а всередині алгоритмів кластеризації абстрагуватись від того, яка конкретно метрика використовується, за допомогою механізму поліморфізму, доступного у C++. В реалізації метрик я зупинився на найпоширенішій на даний момент евклідовій метриці.

Після перших пробних запусків та профілювання я виявив, що найбільше часу витрачається на виконання функції обчислення відстані між об'єктами. Проаналізувавши дані профілювання, я побачив, що левову частку ресурсів займає доступ до атрибутів об'єкта через створену мною структуру даних. Це заставило мене спростити існуючу структуру, видаливши із неї клас Attribute, і залишивши масив чисел, що відповідають атрибутам, як поле класу Object. Інша суттєва проведена мною оптимізація звелась до відмови від використання чисел із плаваючою комою точності (double) та переходу на числа одинарної точності (float). Швидкодія сучасних процесорів при роботі із останніми значно вища, в чому я мав можливість наглядно пересвідчитись.

Наступний етап очевидної оптимізації полягав у відмові від адресації

об'єктів по ідентифікатору та переході від асоціативного контейнера до звичайного. Я виявив, що адресація за ідентифікатором не використовується взагалі, проте на неї витрачається багато ресурсів. Перейшовши до адресації за порядковим номером об'єкта в контейнері std::vector, що відтепер містив вказівники на об'єкти, я зекономив досить багато ресурсів.

Подальші оптимізації в більшості стосувались розпаралелення виконання однотипних завдань. Наприклад, задача пошуку сусідів точки в її ϵ-околі дозволяє розбити процес пошуку на декілька взаємонезалежних потоків, кожен з яких обробляв би певну частину вибірки, і по завершенню їх усіх просто об'єднати отримані дані. Оскільки програма виконувалась на двоядерному процесорі, це дозволило наростити продуктивність.

Написання програми відбувалось у текстовому редакторі vim. Використовувався компілятор gcc. Для відлагодження програми я використовував gdb. Відслідковування витрат пам'яті та профілювання програми здійснювалось за допомогою valgrind. При розробці я використовував систему контролю версій git.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Arthur, D.* K-means has polynomial smoothed complexity / D. Arthur, B. Manthey, H. R. A. // Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2009) / Ed. by D. Spielman. — Los Alamitos: IEEE Computer Society Press, 2009. — Pp. 405–414.
2. *Arthur, D.* How slow is the k-means method? / D. Arthur, S. Vassilvitskii // SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry. — New York, NY, USA: ACM, 2006. — Pp. 144–153.
3. *Davies, D. L.* A cluster separation measure / D. L. Davies, D. W. Bouldin // *IEEE Transactions on Pattern Analysis and Machine Intelligence.* — 1979. — Vol. 1, no. 2. — P. 224–227.
4. A density-based algorithm for discovering clusters in large spatial databases with noise / M. Ester, H.-P. Kriegel, J. S., X. Xu. — AAAI Press, 1996. — Pp. 226–231.
5. *Dunn, J. C.* Well separated clusters and optimal fuzzy partitions / J. C. Dunn // *Journal of Cybernetics.* — 1974. — Vol. 4. — Pp. 95–104.
6. *Eppstein, D.* Fast hierarchical clustering and other applications of dynamic closest pairs. / D. Eppstein // *J. Exp. Algorithmics.* — 2000. — no. 5. — Pp. 1–23.
7. *Hamerly, G.* Alternatives to the k-means algorithm that find better clusterings / G. Hamerly, C. Elkan // CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management. — New York, NY, USA: ACM, 2002. — Pp. 600–607.
8. *MacQueen, J. B.* Some methods for classification and analysis of multivariate observations / J. B. MacQueen // *Proceedings of 5th Berkeley Sympo-*

sium on Mathematical Statistics and Probability. — 1967. — P. 281–297.

9. Np-hardness of euclidean sum-of-squares clustering / D. Aloise, A. Deshpande, P. Hansen, P. Popat // *Machine Learning.* — 2009. — no. 75. — Pp. 245–249.
10. *Rosenblatt, F.* The perceptron – a perceiving and recognizing automaton: Tech. Rep. 85-460-1 / F. Rosenblatt: Cornell Aeronautical Laboratory, 1957.
11. *Socal, R.* Principles of Numerical Taxonomy / R. Socal, P. Sneath. — San Francisco: W.H. Freeman, 1963.
12. *Sokal, R.* A statistical method for evaluating systematic relationships / R. Sokal, C. Michener // *University of Kansas Science Bulletin.* — 1958. — Vol. 38. — Pp. 1409–1438.
13. *Spielman, D.* Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time / D. Spielman, S. H. Teng // STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing. — New York, NY, USA: ACM, 2001. — Pp. 296–305.
14. *Steinhaus, H.* Sur la division des corps matériels en parties / H. Steinhaus // *Bulletin of the Polish academy of Sciences.* — 1956. — Vol. 3, no. 4. — Pp. 801–804.
15. *Tryon, R. C.* Cluster Analysis / R. C. Tryon. — Ann Arbor: Edwards Brothers, 1939.
16. *Загоруйко.* Прикладные методы анализа данных и знаний / Загоруйко. — Издательство института математики, Новосибирск, 1999.
17. *Терентьев П. В.* Метод корреляционных плеяд / Терентьев П. В. — Вестник Ленинградского университета, 1959.